Two for One: SerDes Flows for AMI Model Development

Corey Mathis, Ren Sang Nah (MathWorks)
Richard Allred, Todd Westerhoff (SiSoft)

DesignCon 2016 IBIS Summit
Santa Clara, California
January 22, 2016

* Adapted from the DesignCon 2016 paper “Leveraging SerDes Flows for AMI Model Development”
Background

• Good AMI models are hard to develop
  – Analog / algorithmic partitioning
  – IBIS-AMI requirements: samples per bit
  – Portability issues between EDA tools

• AMI development typically occurs “after the fact”
  – AMI Models are “customer collateral”
  – Created by different group
  – Limited testing before distribution
SerDes Design

• Architectural models
  – Created up-front to define architecture & budgets
  – Limited design detail, execute relatively fast
  – Good for design budgets & control loop behavior

• Implementation models
  – Detail varies from architectural to gate to circuit level
  – A “snapshot” of the design at a point in time

• This presentation is based on Architectural models
An AMI Model Primer

Fundamental Assumption

Model Components
# AMI Simulation Primer

## Algorithmic Model

- **AMI_Init**
  - Model configuration parameters
  - Impulse response processing
  - Linear, Time-Invariant (LTI)

- **AMI_Getwave**
  - Waveform processing
  - Clock ticks
  - Non-Linear, Time-Varying (NLTV)

- **AMI_Close**
  - Clean up & exit

## Channel Simulation

- Network Characterization (Circuit Simulation)
- Channel Simulation (Signal Processing)

For more info: “Understanding IBIS-AMI Simulations”, DesignCon 2015
SerDes Architectural Exploration

Typical flow: end to end link in Architectural simulator

Our flow: leverage strengths of Architectural and AMI simulators
IBIS-AMI Model Development Loop

AMI Simulator
- Analog model
- Test channels
- Test case generation
- Batch & regression tests
- Generate test setups

Architectural Simulator
- Interactive design and debug
- Export AMI .dll and .ami files
- Identify cases of interest
- Batch testing – many channels & model settings

AMI Simulator
- Interactive design & analysis
- Test case debugging
Leveraging Existing Infrastructure

Build on existing capabilities for embedded software code generation
Mixing Structure and Code

• Key parts of SerDes designs are often implemented as “code”
• Ability to mix structure and code is critical
Creating Algorithmic Models

Identify model type
- Identify AMI parameters
  - Based on design characteristics (Init/LTI or Getwave/NLTV)

Generate Code & Compile
- C++ code
- .ami file
IBIS-AMI Model Types

TX: 4 tap filter

Simple, linear, non-adaptive ➔ “Init” or LTI model

RX: CTLE, Saturation, 8 tap DFE, CDR

Complex, non-linear, adaptive ➔ “Getwave” or NLTV model

Design characteristics drive proper IBIS-AMI model type
A Simple AMI Transmitter

4 tap TX with normalization captured two ways

Structural model

Channel behavior with & without equalization

MATLAB code
Two for One: SerDes Flows for AMI Model Development

Generated C++ Code

AMI wrapper

Model behavior
Architectural vs. AMI Results

Architectural simulation ends, AMI simulation run for more bits

Test bench

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX output resistance</td>
<td>50 ohms</td>
</tr>
<tr>
<td>TX output capacitance</td>
<td>0.5 pF</td>
</tr>
<tr>
<td>TX output voltage supply</td>
<td>1 volt</td>
</tr>
<tr>
<td>TX output rise/fall time</td>
<td>10 ps</td>
</tr>
<tr>
<td>TX EQ applied</td>
<td>Cursor = 0.715, \text{i}^{th} Postcursor = -0.285</td>
</tr>
<tr>
<td>Data rate</td>
<td>100ps</td>
</tr>
<tr>
<td>Channel model</td>
<td>Simple lossy stripline</td>
</tr>
<tr>
<td>RX input resistance</td>
<td>50 ohms</td>
</tr>
<tr>
<td>RX input capacitance</td>
<td>Zero</td>
</tr>
<tr>
<td>RX EQ applied</td>
<td>None</td>
</tr>
</tbody>
</table>

Output waveform comparison
USB 3.0 Receiver Model

- Low power mobile receiver with multi-protocol support
- Design challenge: balance AGC, linear/non-linear EQ and CDR
- USB3.0 On-the-Go (OTG) support especially challenging
Model Correlation

Simulator to Simulator

Architectural Model

AMI Model

Simulator to Hardware

Receiver hardware reports eye height and width based on sampling clock
Other Findings

- Expose “extra” controls and outputs
- Internal testing & design tuning
- Internal regression testing

- Fewer exposed controls & outputs
- Early models for key customer feedback
Other Findings

Throughput

- AMI models run 4-8x faster than their Architectural counterparts
- Running AMI simulations in parallel can provide ~150x speedup for regression testing

AMI Compliance

- AMI requires models run at any setting of “samples per bit”
- Architectural models can be set up to meet this requirement
AMI RX Model

AMI Parameters

Saturation Block Code, Continuous Time, Time-Domain

Peaking Filter
Structural, Continuous Time, Frequency-Domain

CDR/DFE Code, Sampled Time, Time-Domain

Clock Ticks output
AMI RX - Architectural Simulation

- Data Clock Position
- CDR Inc./Dec.
- Early / Late Count
- RX @ Pad
- RX @ Latch
- DFE Taps
AMI RX – Compiled Model

- Comparable to hand-written models
Summary

• AMI models can be created from Architectural models normally created during the SerDes design cycle
• The parameters exposed in an AMI model can be varied depending on the application
• Models produced with this process behave just like any other well-constructed AMI model